Data Visualization

Fall 2024
Fridays 9am - 11am
Avery 409

Week 7

1. Basics of geospatial data
2. A tour of mapbox
3. Final project groups

Geospatial Data

Definitions

Resources

Projects

Tools and Workflows

Geospatial Data

Definitions

Resources

Projects

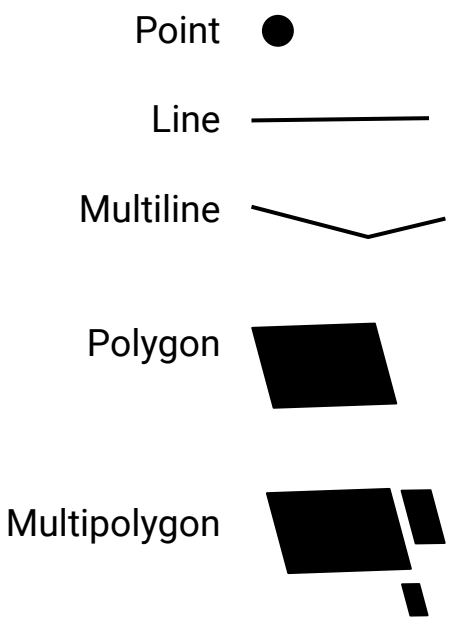
Tools and Workflows

Geospatial Data

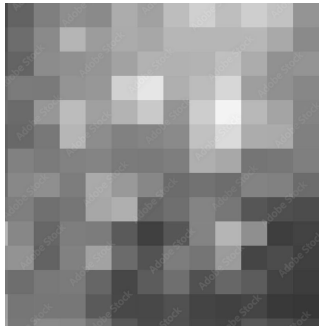
Definitions

2 types

Vector data is
Discrete
and made of features



Raster data is
Continuous
and made of cells



Geospatial Data

Definitions

2 types

Vector data is
Discrete
and made of features

Point

Line

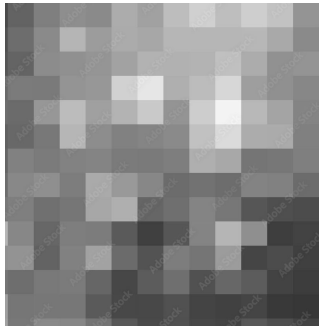
Multiline

Polygon

Multipolygon

Each feature has an attribute table

Raster data is
Continuous
and made of cells



Each cell has a value

Geospatial Data

We most often use geospatial data in the geojson file format

Geographical + JSON = geojson

JavaScript
Object
Notationt

Geospatial Data

JSON consists of key and value pairs. Here is an example.

```
{
    "key": "value"
}
```

Geospatial Data

JSON consists of key and value pairs. Here is another example.

```
{
   "glossary": {
          "title": "example glossary",
          "GlossDiv": {
                 "title": "S",
                 "GlossList": {
                        "GlossEntry": {
                               "ID": "SGML",
                               "SortAs": "SGML",
                               "GlossTerm": "Standard Generalized Markup Language",
                               "Acronym": "SGML",
                               "Abbrev": "ISO 8879:1986",
                               "GlossDef": {
                                      "para": "A meta-markup language, used to create markup languages such as DocBook.",
                                      "GlossSeeAlso": ["GML", "XML"]
                               },
                               "GlossSee": "markup"
                        }
                 }
          }
   }
}
```

Geospatial Data

JSON consists of key and value pairs. Here is another example.

```
{
   "glossary": {
          "title": "example glossary",
          "GlossDiv": {
                 "title": "S",
                 "GlossList": {
                        "GlossEntry": {
                               "ID": "SGML",
                               "SortAs": "SGML",
                               "GlossTerm": "Standard Generalized Markup Language",
                               "Acronym": "SGML",
                               "Abbrev": "ISO 8879:1986",
                               "GlossDef": {
                                      "para": "A meta-markup language, used to create markup languages such as DocBook.",
                                      "GlossSeeAlso": ["GML", "XML"]
                               },
                               "GlossSee": "markup"
                        }
                 }
          }
   }
}
```

Geospatial Data

Vector Data JSON format example:


```
{
"type": "FeatureCollection",
"name": "prison_boundaries",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::4269" } },
"features": [
{ "type": "Feature", "properties": { "FID": 1, "FACILITYID": "10002798", "NAME": "MIDLAND
COUNTY CENTRAL DETENTION CENTER", "ADDRESS": "400 S MAIN ST", "CITY": "MIDLAND", "STATE":
"TX", "ZIP": "79701", "ZIP4": "-999", "TELEPHONE": "(432) 688-4745", "TYPE": "COUNTY",
"STATUS": "OPEN", "POPULATION": 438, "COUNTY": "MIDLAND", "COUNTYFIPS": "48329",
"COUNTRY": "USA", "NAICS_CODE": "922140", "NAICS_DESC": "CORRECTIONAL INSTITUTIONS",
"SOURCE": "https://www.co.midland.tx.us/200/Detention", "SOURCEDATE": "2018-04-30",
"VAL_METHOD": "IMAGERY/OTHER", "VAL_DATE": "2020-02-27", "WEBSITE":
"https://www.co.midland.tx.us/departments/SO/Pages/default.aspx", "SECURELVL": "MAXIMUM",
"CAPACITY": 498, "SHAPE_Leng": 0.0072747441475100004, "GlobalID":
"{86DE7B8A-37D4-4D99-984B-D70D2B8C474F}", "CreationDa": "2022-01-07", "Creator":
"HostedByHIFLD", "EditDate": "2022-01-07", "Editor": "HostedByHIFLD", "SHAPE_Le_1":
863.8710760592603, "SHAPE_Area": 2.579058339e-06 }, "geometry": { "type": "MultiPolygon",
"coordinates": [ [ [ [ -102.076874037287752, 31.993805467817296 ], [
-102.074421982413483, 31.995045048990676 ], [ -102.074425552318431, 31.993406790076719 ],
[ -102.076838324763628, 31.993331837340406 ], [ -102.076874037287752, 31.993805467817296
] ] ] ] } },...]}
```

Geospatial Data

Vector Data JSON format example:

This is a feature collection


```
{
"type": "FeatureCollection",
"name": "prison_boundaries",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::4269" } },
"features": [
{ "type": "Feature", "properties": { "FID": 1, "FACILITYID": "10002798", "NAME": "MIDLAND
COUNTY CENTRAL DETENTION CENTER", "ADDRESS": "400 S MAIN ST", "CITY": "MIDLAND", "STATE":
"TX", "ZIP": "79701", "ZIP4": "-999", "TELEPHONE": "(432) 688-4745", "TYPE": "COUNTY",
"STATUS": "OPEN", "POPULATION": 438, "COUNTY": "MIDLAND", "COUNTYFIPS": "48329",
"COUNTRY": "USA", "NAICS_CODE": "922140", "NAICS_DESC": "CORRECTIONAL INSTITUTIONS",
"SOURCE": "https://www.co.midland.tx.us/200/Detention", "SOURCEDATE": "2018-04-30",
"VAL_METHOD": "IMAGERY/OTHER", "VAL_DATE": "2020-02-27", "WEBSITE":
"https://www.co.midland.tx.us/departments/SO/Pages/default.aspx", "SECURELVL": "MAXIMUM",
"CAPACITY": 498, "SHAPE_Leng": 0.0072747441475100004, "GlobalID":
"{86DE7B8A-37D4-4D99-984B-D70D2B8C474F}", "CreationDa": "2022-01-07", "Creator":
"HostedByHIFLD", "EditDate": "2022-01-07", "Editor": "HostedByHIFLD", "SHAPE_Le_1":
863.8710760592603, "SHAPE_Area": 2.579058339e-06 }, "geometry": { "type": "MultiPolygon",
"coordinates": [ [ [ [ -102.076874037287752, 31.993805467817296 ], [
-102.074421982413483, 31.995045048990676 ], [ -102.074425552318431, 31.993406790076719 ],
[ -102.076838324763628, 31.993331837340406 ], [ -102.076874037287752, 31.993805467817296
] ] ] ] } },...]}
```

Geospatial Data

Vector Data JSON format example:

This is one of the features


```
{
"type": "FeatureCollection",
"name": "prison_boundaries",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::4269" } },
"features": [
{ "type": "Feature", "properties": { "FID": 1, "FACILITYID": "10002798", "NAME": "MIDLAND COUNTY CENTRAL DETENTION CENTER", "ADDRESS": "400 S MAIN ST", "CITY": "MIDLAND", "STATE": "TX", "ZIP": "79701", "ZIP4": "-999", "TELEPHONE": "(432) 688-4745", "TYPE": "COUNTY", "STATUS": "OPEN", "POPULATION": 438, "COUNTY": "MIDLAND", "COUNTYFIPS": "48329", "COUNTRY": "USA", "NAICS_CODE": "922140", "NAICS_DESC": "CORRECTIONAL INSTITUTIONS", "SOURCE": "https://www.co.midland.tx.us/200/Detention", "SOURCEDATE": "2018-04-30", "VAL_METHOD": "IMAGERY/OTHER", "VAL_DATE": "2020-02-27", "WEBSITE": "https://www.co.midland.tx.us/departments/SO/Pages/default.aspx", "SECURELVL": "MAXIMUM", "CAPACITY": 498, "SHAPE_Leng": 0.0072747441475100004, "GlobalID": "{86DE7B8A-37D4-4D99-984B-D70D2B8C474F}", "CreationDa": "2022-01-07", "Creator": "HostedByHIFLD", "EditDate": "2022-01-07", "Editor": "HostedByHIFLD", "SHAPE_Le_1": 863.8710760592603, "SHAPE_Area": 2.579058339e-06 }, "geometry": { "type": "MultiPolygon", "coordinates": [ [ [ [ -102.076874037287752, 31.993805467817296 ], [ -102.074421982413483, 31.995045048990676 ], [ -102.074425552318431, 31.993406790076719 ], [ -102.076838324763628, 31.993331837340406 ], [ -102.076874037287752, 31.993805467817296 ] ] ] ] } },...]}
```

Geospatial Data

Vector Data JSON format example:

JSON format syntax

{

Key and value pairs that are attributes of the feature collection,

Features: [list of features with attributes]

}

Geospatial Data
Definitions

*As all data, geospatial data always has the the **dimension of time**!*

1. When data was last collected
2. Interval between data collections
3. History of data collection

Geospatial Data

Definitions

Resources

Projects

Tools and Workflows

Geospatial Data Sources

Resources

Columbia
https://geodata.library.columbia.edu/catalog?f%5Bdc_subject_sm%5D%5B%5D=Boundaries

Open Street Maps
https://www.openstreetmap.org/#map=4/38.01/-95.84

data.gov
https://catalog.data.gov/dataset/?metadata_type=geospatial

census.gov
https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html

USGS data gateway
https://www.usgs.gov/products/data/all-data

Homeland Infrastructure foundation-level data
https://hifld-geoplatform.opendata.arcgis.com/search?collection=Dataset

Geospatial Data

Definitions

Resources

Projects

Tools and Workflows

Geospatial Data

Projects fall into one or more of these groups:

Analysis - presenting new knowledge and insight in map form

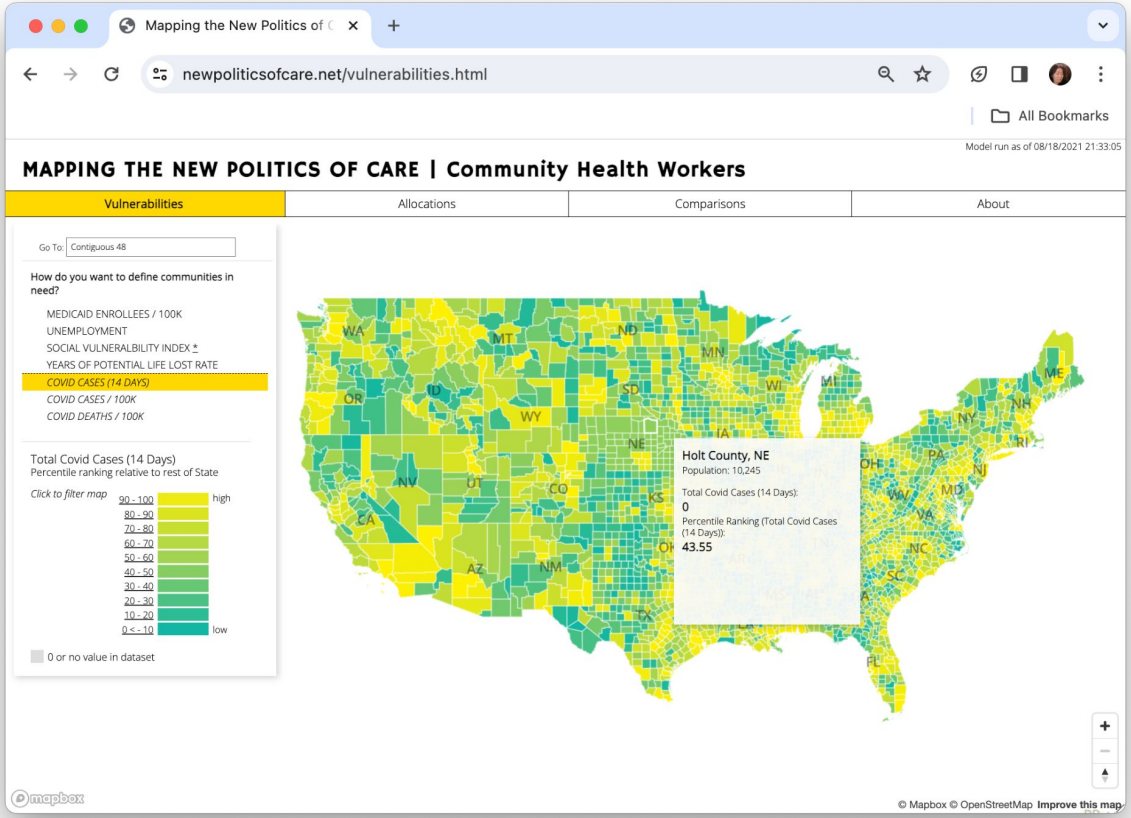
Basemaping - Creating and publishing critical data for reuse

Narrative - Highlighting research with storytelling to reach a wider audience

Qualitative - contextualizing dataset with qualitative information such as images.

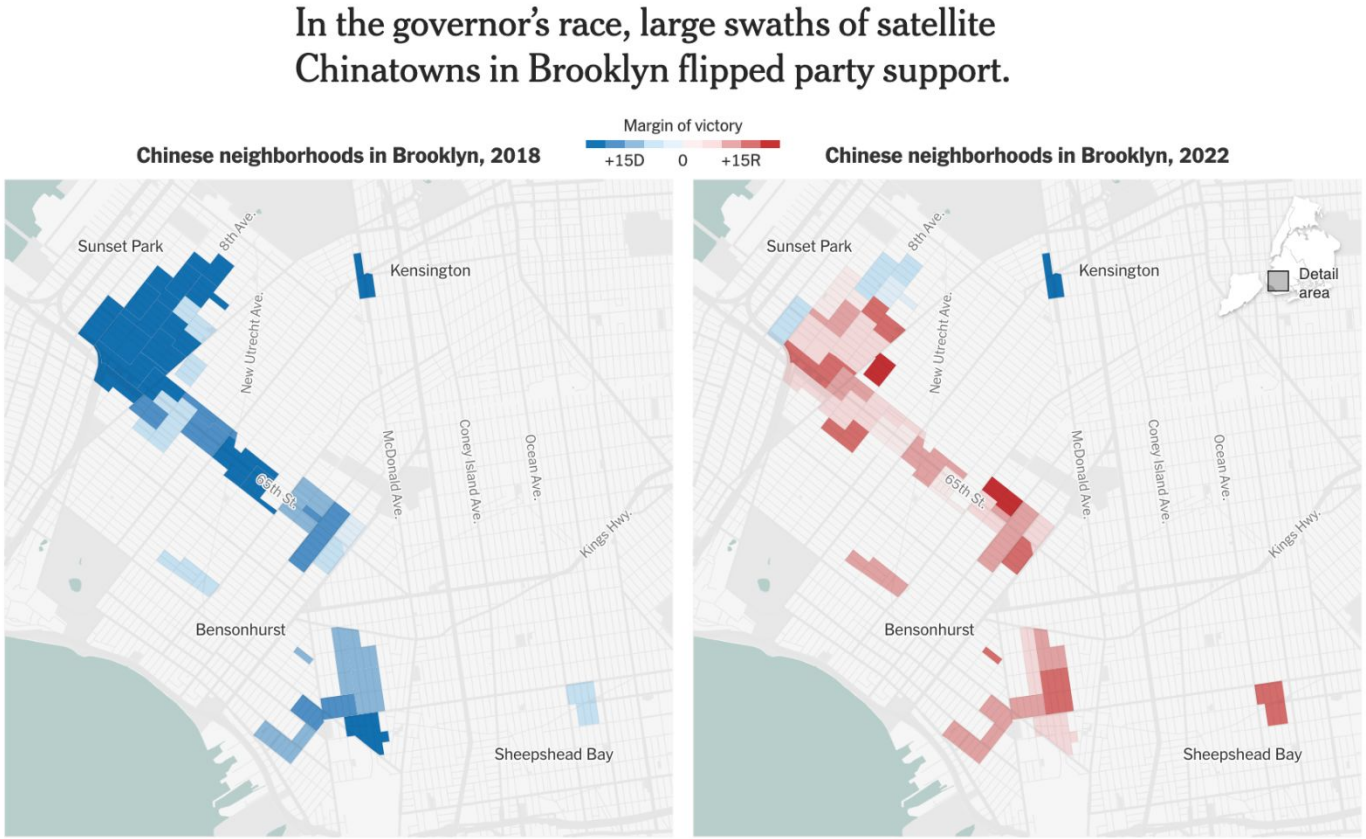
Location driven - uses gps as the mechanism to filter and communicate data

Geospatial and other analysis - presenting new knowledge and insight in map form

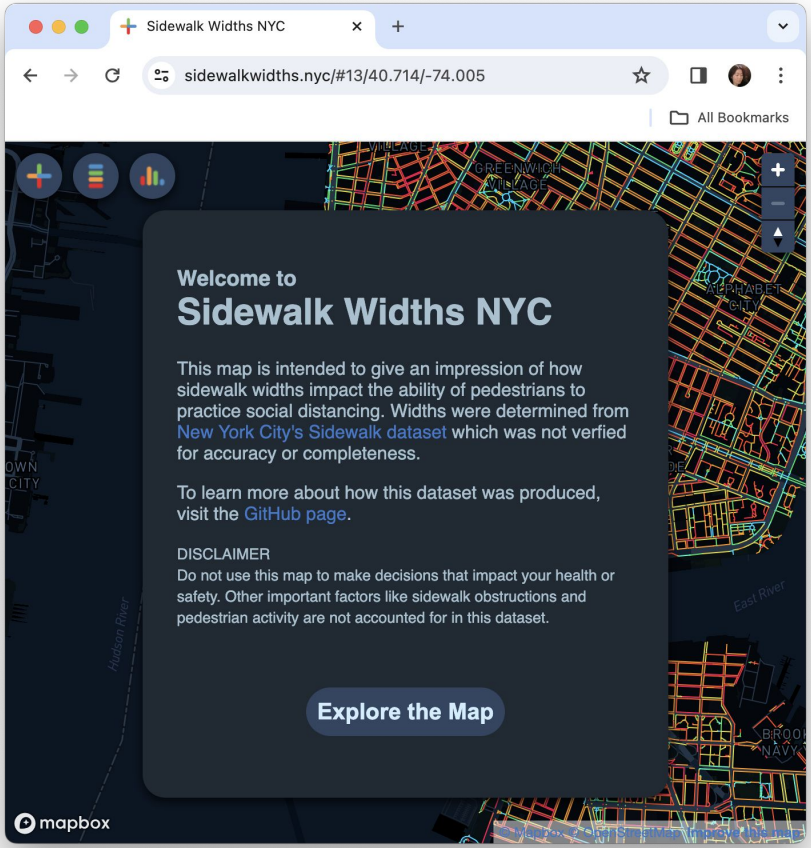


Geospatial and other analysis - presenting new knowledge and insight in map form

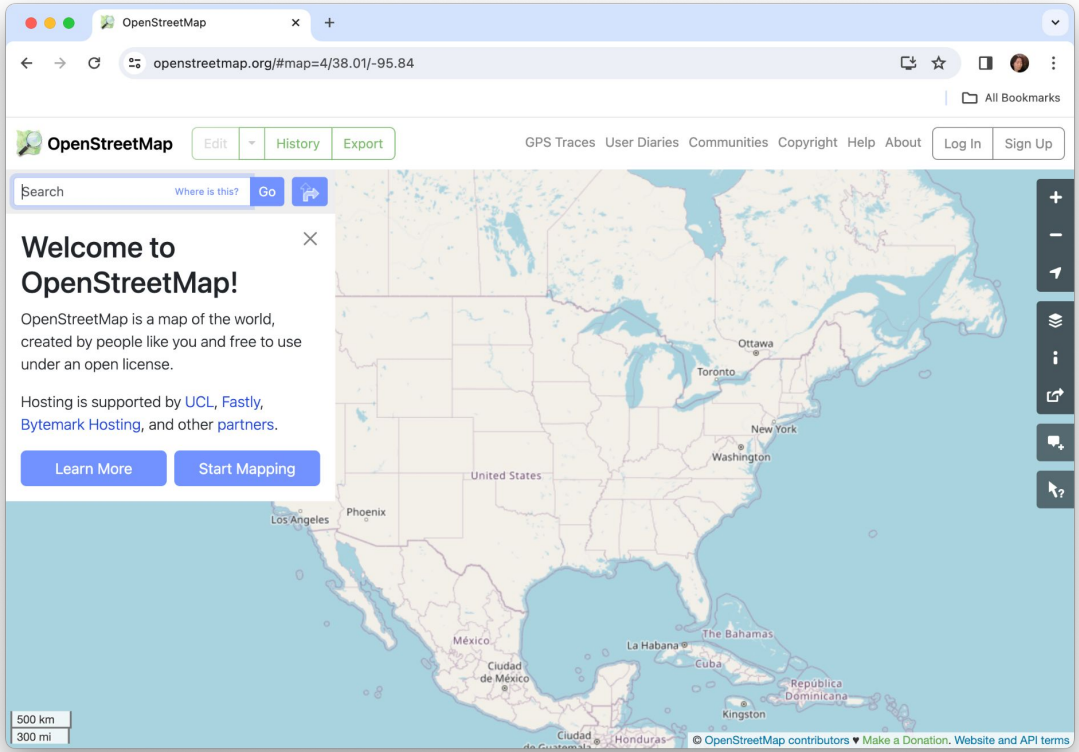
NEW YORK | Where New York's Asian Neighborhoods Shifted to the Right



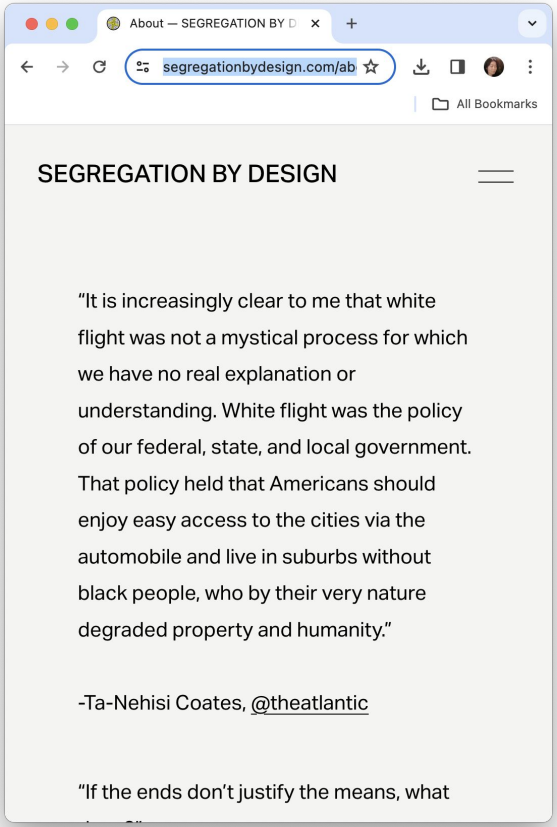
Geospatial and other analysis - presenting new knowledge and insight in map form



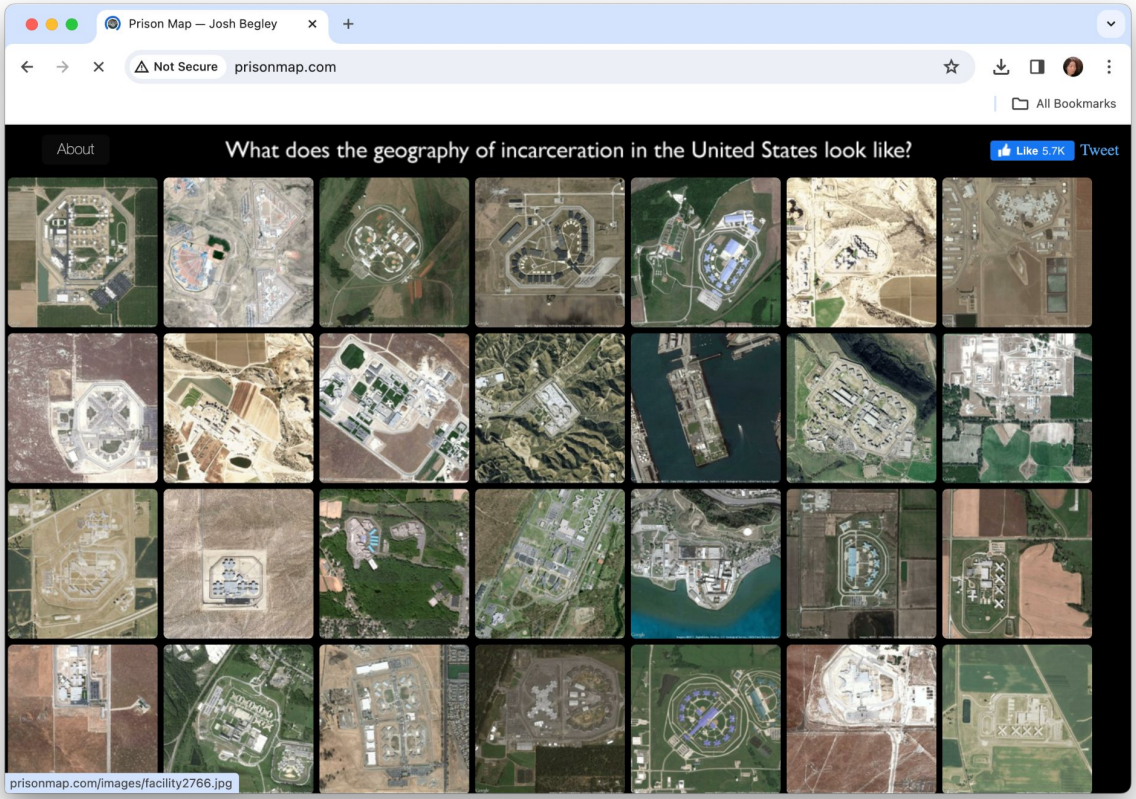
Basemaping - Creating and publishing critical data for reuse



Qualitative - contextualizing dataset with qualitative information such as images.



Qualitative - contextualizing dataset with qualitative information such as images.



Geospatial Data

Definitions

Resources

Projects

Tools and Workflows

Mapbox basics

Style
Setting map bounds
Adding control for zoom
Geolocating the user
Search function
Tilt
Directions
Icons
Flying
Layers
External data
Mapbox credit

Compare map
Details map

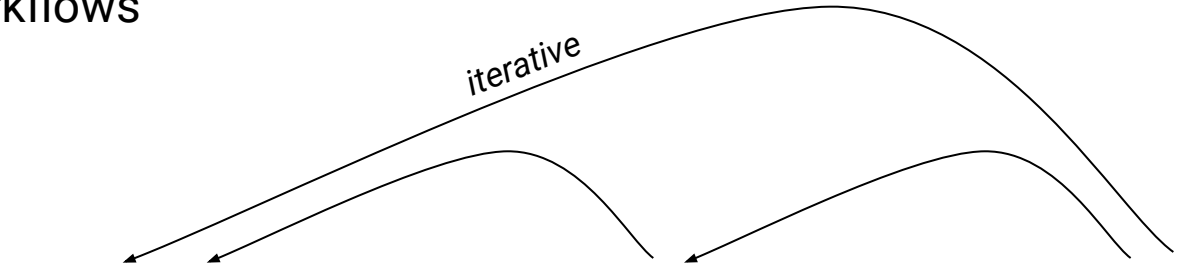
For next week

1. Form and finalize groups
2. Choose a map tutorial to complete

Next week

3. Scrollytelling for maps
4. Scrollytelling for charts

Tools and Workflows



A general workflow

Data Production
Acquiring the right data

→

Process
Identify value

→

Presentation
Communicate publicly

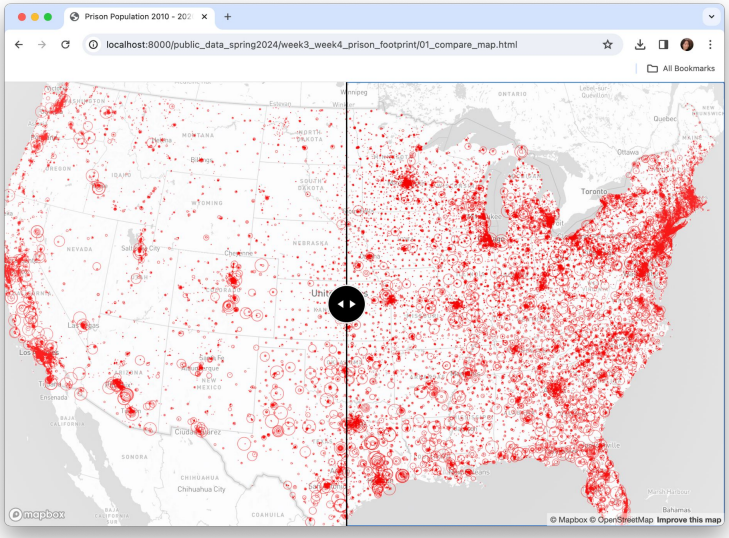
Examples of technologies used at each stage

Data Production	Process	Presentation
Portals	Python	Writing
FOIA	R	Mapbox
Scrappers	GIS	javascript

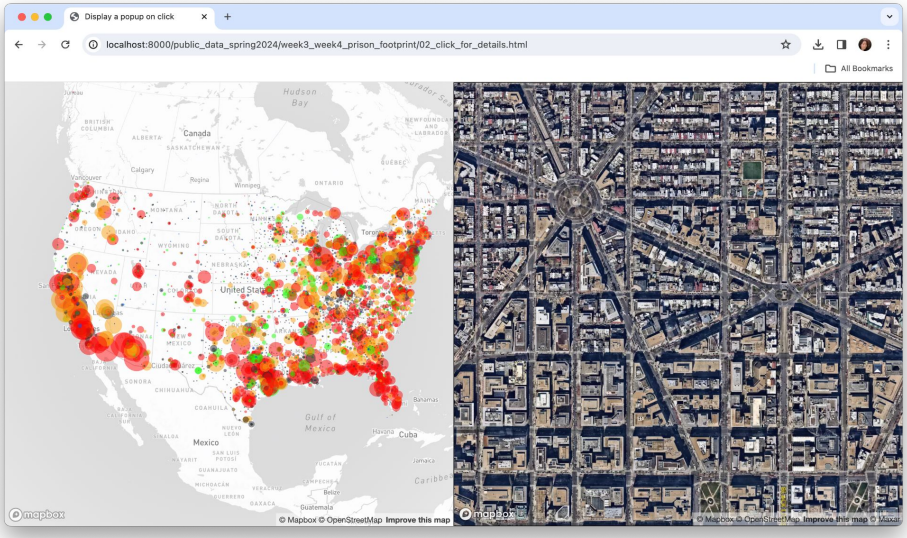
Project Example:

2 quick experiments in the geography of U.S. incarceration

Comparisons for change



Click to show details



Tools and Workflows

Tools:

- QGIS
- Mapbox
- Text editor
- Local Server

Files:

Interactive map files:

- comparison.html
- scroll.html
- clicks.html

Data files:

- Prison footprints: https://hifld-geoplatform.opendata.arcgis.com/datasets/geoplatform::prison-boundaries
- 2020 - https://www.prisonersofthecensus.org/data/2020/group_quarters_shapefile.html
- 2010 - https://www.prisonersofthecensus.org/data/2010/groupquartersshapefile.html

D3 maps

Maps

Basemap

```
<script type="text/javascript">
//set up svg, nothing new here
var width = window.innerWidth
var height = window.innerHeight
var svg = d3.select("body")
    .append("svg")
    .attr("width",width)
    .attr("height", height);

//we need a data file, and a geopath file
var dataPath = "NYC_Parks_Events_2019.csv";//these are some park events with coord
var geoPath = "nyc_boroughs.geojson";//this is a geojson, which is json format of

//promise allows us to load many files, and only excutes the code inside it once t
//before we used d3.csv to load 1 csv file and .then to excute code once its been
Promise.all([d3.json(geoPath), d3.csv(dataPath)])//here we use d3.json to load the
.then(function(data) {//.then is the same as before, we are just saying wait till
    var geo = data[0];// data in blue in the line above is the result of the promi
    var places = data[1];//and the second is the place markers
    //console.log(data)//try uncommenting and seeing what data is at this point to

    drawOutline(geo) //call the draw outline function from below
});

function drawOutline(geo){
    var padding = 50
    var projection = d3.geoAlbers()
            .fitExtent([[padding,padding],[width-padding,height-padding]],geo)

    var path = d3.geoPath().projection(projection);

    svg.append("path")
        .attr("d", path(geo))
        .attr("fill", "none")
        .attr("stroke", "#aaaaaa")
        .attr("stroke-width",1)
}
</script>
```

Maps

Basemap

3 parts
Only last part draws map

Svg stuff you have seen similar code before

Load 2 files, the boundaries and the data

Draws the outlines

```
<script type="text/javascript">
//set up svg, nothing new here
var width = window.innerWidth
var height = window.innerHeight
var svg = d3.select("body")
    .append("svg")
    .attr("width",width)
    .attr("height", height);

//we need a data file, and a geopath file
var dataPath = "NYC_Parks_Events_2019.csv";//these are some park events with coord
var geoPath = "nyc_boroughs.geojson";//this is a geojson, which is json format of

//promise allows us to load many files, and only excutes the code inside it once t
//before we used d3.csv to load 1 csv file and .then to excute code once its been
Promise.all([d3.json(geoPath), d3.csv(dataPath)])//here we use d3.json to load the
.then(function(data) {//.then is the same as before, we are just saying wait till
    var geo = data[0];// data in blue in the line above is the result of the promi
    var places = data[1];//and the second is the place markers
    //console.log(data)//try uncommenting and seeing what data is at this point to

    drawOutline(geo) //call the draw outline function from below
});

function drawOutline(geo){
    var padding = 50
    var projection = d3.geoAlbers()
            .fitExtent([[padding,padding],[width-padding,height-padding]],geo)

    var path = d3.geoPath().projection(projection);

    svg.append("path")
        .attr("d", path(geo))
        .attr("fill", "none")
        .attr("stroke", "#aaaaaa")
        .attr("stroke-width",1)
}
</script>
```

Maps

Finally drawing the map

Steps: this will be similar to line graph from before

1. ***Define a projection***
2. ***Define a path***
3. ***Append the path***
4. ***Set path to definition***
5. ***Style***

```
function drawOutline(geo){
    var padding = 50
    var projection = d3.geoAlbers()
            .fitExtent([[padding,padding],[width-padding,height-padding]],geo)

    var path = d3.geoPath().projection(projection);

    svg.append("path")
        .attr("d", path(geo))
        .attr("fill", "none")
        .attr("stroke", "#aaaaaa")
        .attr("stroke-width",1)
}
```

Define projection

Define path

Append
Set to definition

Style

Maps

Finally drawing the map

The new thing you need is the projection. There are many projections in d3. We are using albers.

```
function drawOutline(geo){
    var padding = 50
    var projection = d3.geoAlbers()
            .fitExtent([[padding,padding],[width-padding,height-padding]],geo)

    var path = d3.geoPath().projection(projection);

    svg.append("path")
        .attr("d", path(geo))
        .attr("fill", "none")
        .attr("stroke", "#aaaaaa")
        .attr("stroke-width",1)
}
```

Maps

Finally drawing the map

A projection also needs a center, and a scale. Here we are sidestepping that by saying let's fit the geo(the geojson data loaded) into the width and height of the svg. (minus padding)

```
function drawOutline(geo){
    var padding = 50
    var projection = d3.geoAlbers()
            .fitExtent([[padding,padding],[width-padding,height-padding]],geo)

    var path = d3.geoPath().projection(projection);

    svg.append("path")
        .attr("d", path(geo))
        .attr("fill", "none")
        .attr("stroke", "#aaaaaa")
        .attr("stroke-width",1)
}
```

Maps

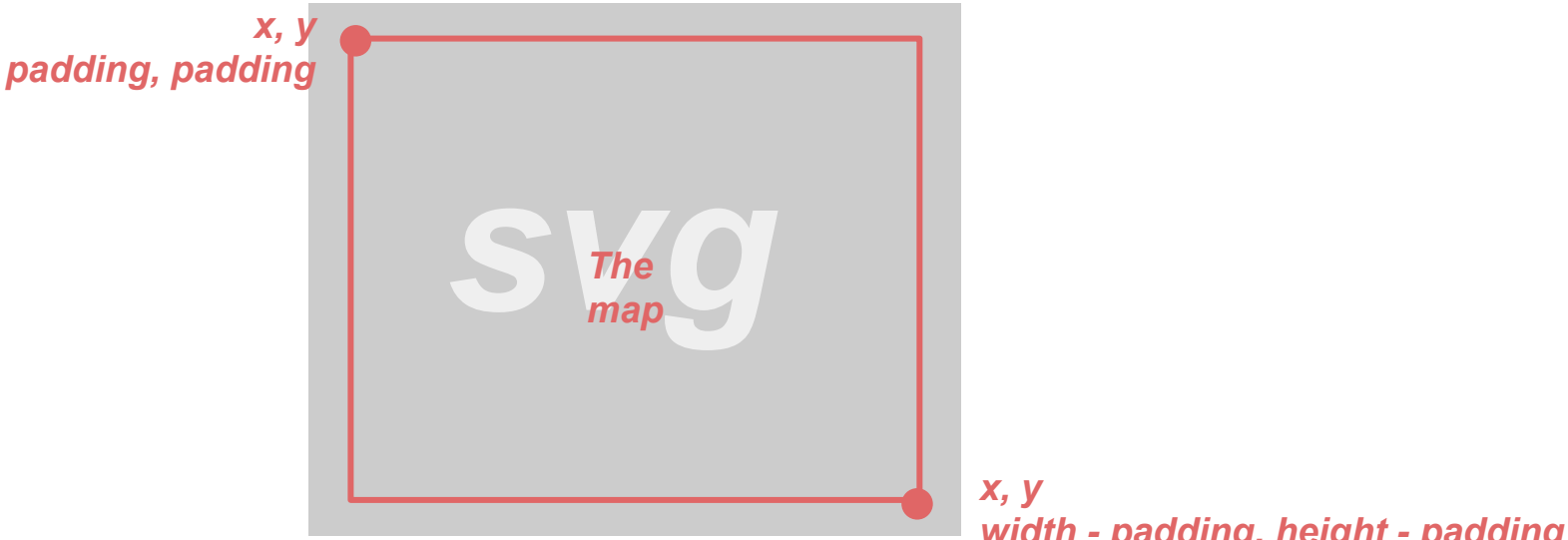
Finally drawing the map

```
.fitExtent([[padding,padding],[width-padding,height-padding]],geo)
```

Maps

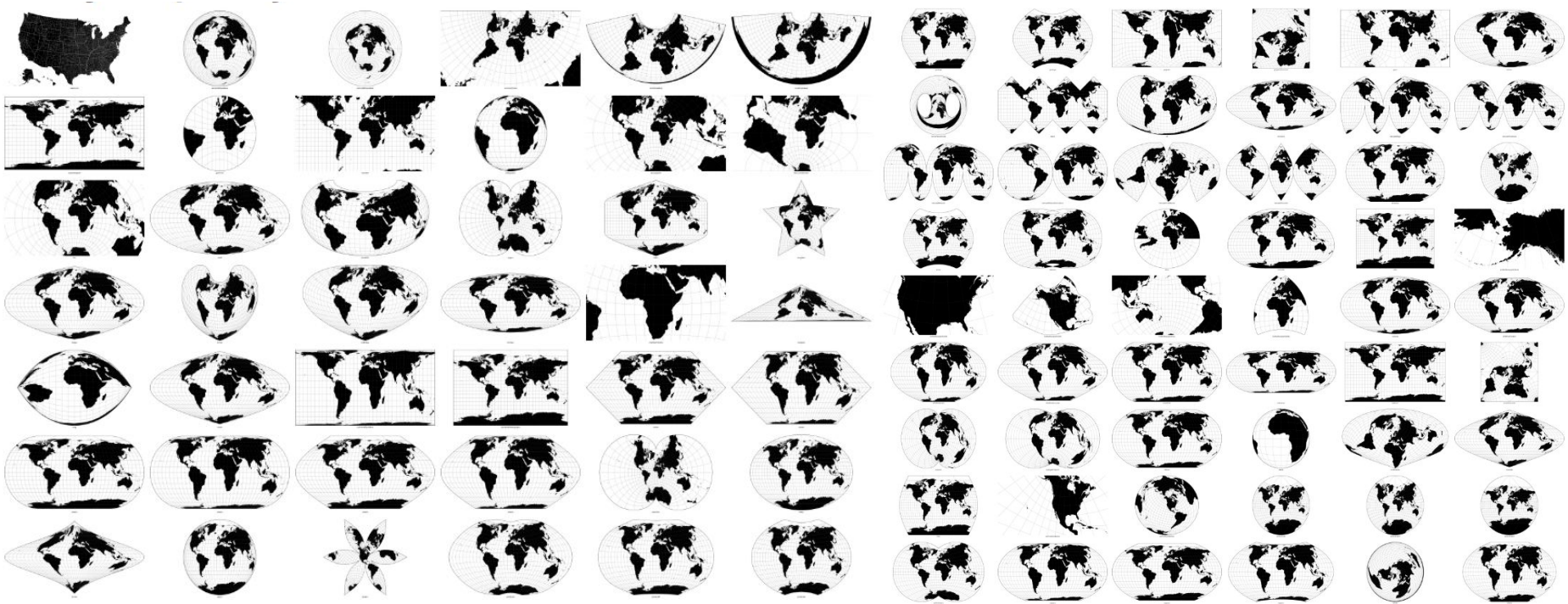
Finally drawing the map

```
projection = d3.geoAlbers()
  .fitExtent([[padding,padding],[width-padding,height-padding]],geo)
```



All the Projections

https://bl.ocks.org/mbostock/29cddc0006f8b98eff12e60dd08f59a7



Maps

Basemap

```
function drawOutline(geo){
    var padding = 50
    var projection = d3.geoAlbers()
            .fitExtent([[padding,padding],[width-padding,height-padding]],geo)

    var path = d3.geoPath().projection(projection);

    svg.append("path")
        .attr("d", path(geo))
        .attr("fill", "none")
        .attr("stroke", "#aaaaaa")
        .attr("stroke-width",1)
}
```

Maps

Top layer

Just drawing circles where "cx" and "cy" are the projected lattitude and longitude

```
svg.selectAll("circle")
    .data(places)
    .enter()
    .append("circle")
    .attr("r", 5)
    .attr("cx", function(d) {
        return projection([d.long,d.lat])[0]
    })
    .attr("cy", function(d) {
        return projection([d.long,d.lat])[1]
    })
    .attr("opacity", 0.1)
```

Maps

Top layer

```
    .attr("cx", function(d) {
        return projection([d.long,d.lat])[0]
    })
```

Projects an array [longitude, latittude] To an array [x,y] in terms of the window

To get cx from array of [x,y], you get the first item, using index [0]

If time allows, log out in code: ***projection([lng, lat])***